МИНИСТЕРСТВО НАУКИ И ВЫСШЕГООБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики и кибернетики
Кафедра технической кибернетики

Отчёт по предмету «Технологии программирования на Python»

Обучающийся _____ С.А. Прохоров
(*подпись*)

Преподаватель _____ Н.В.Головастиков
(*подпись*)

Самара 2025

СОДЕРЖАНИЕ

1 Введение

Целью данной работы является создание приложения для обработки изображений с использованием внешнего API, освоение современных технологий и методов программирования на Python, включая:

1. Алгоритмы компьютерного зрения (свёртка, выделение границ, детекция углов)
2. Асинхронное программирование и многопроцессорность
3. Работу с внешними API
4. Логирование и тестирование приложений
5. Организацию кода в виде пакета

Для создания приложения были выполнены лабораторные работы 1-5, кратко их требования описаны далее.

1.1 Первое задание

Необходимо:

Реализовать консольное приложение для обработки изображений

1. Реализовать самостоятельно следующие функции: свёртку, приведение цветного изображения к полутоновому, гамма-коррекцию, выделение границ (оператор Собеля), выделение углов (детектор Харриса)
2. Выполнить визуальное сравнение с готовыми функциями для проверки корректности
3. Добавить подсчёт времени выполнения операций

1.2 Второе задание

Необходимо:

1. Подключиться к Cat API (https://thecatapi.com/) и получить API_KEY
2. Скачивать изображения животных с информацией о породе
3. Преобразовывать изображения в numpy-массив и выполнять выделение контуров пользовательским и библиотечным методами

4. Сохранять исходные и обработанные изображения в отдельную поддиректорию с порядковыми номерами
5. Инкапсулировать функционал в двух классах:
    a. CatImage: хранит изображение и метаданные, методы обработки, перегрузка операторов сложения/вычитания/преобразования в строку
    b. CatImageProcessor: работа с API, управление обработкой и сохранением, декоратор для замера времени методов

1.3 Третье задание

Необходимо:

1. Написать скрипт, анализирующий данные в csv-файле в соответствии с вариантом.
2. Каждый отдельный этап обработки (чтение файла, извлечение данных, агрегация) должен осуществляться в отдельном генераторе. Генераторы должны быть организованы в пайплайн.
3. Вывести результаты обработки в виде графика с помощью пакета matplotlib.

1.4 Четвертое задание

Необходимо:

1. Реализовать асинхронное скачивание и сохранение изображений (библиотеки aiohttp и aiofiles)
2. Реализовать параллельную обработку изображений с помощью свёртки в отдельных процессах (multiprocessing или ProcessPoolExecutor)
3. Порядковый номер изображения определяется в начале при получении списка URL и не меняется
4. Добавить замер времени работы программы и вывод информации об этапах обработки с указанием PID процессов

1.5 Пятое задание

Необходимо:

1. Добавить логирование с использованием модуля logging:
    a. В файл app.log (уровень DEBUG с подробной информацией: время, файл, строка)
    b. В консоль (уровень INFO с краткой информацией)
2. Создать модуль тестирования с использованием unittest, реализовать минимум 2 TestCase для проверки функционала CatImage и CatImageProcessor

2 Основная часть

Проект организован в виде модульной структуры и состоит из компонентов, описание которых приведено далее.

2.1 Модуль cat_image.py

2.1.1 Класс CatImage

В Dataclass для инкапсуляции изображения и его метаданных. Содержит поля: image (numpy-массив), url (ссылка на изображение), index (порядковый номер). Реализует перегрузку операторов сложения, вычитания и преобразования в строку.

2.2 Модуль image_processing.py

2.2.1 Класс ImageProcessing

Класс с методами обработки изображений. Содержит реализации алгоритмов компьютерного зрения:

1. Свёртка изображения (_convolution, _convolution2d)
2. Преобразование RGB в grayscale (_rgb_to_grayscale)
3. Гамма-коррекция (_gamma_correction)
4. Выделение границ методом Собеля (edge_detection) и Canny (edge_detection2)
5. Детекция углов методом Харриса (corner_detection, corner_detection2)

2.3 Модуль cat_image_processor.py

2.3.1 Класс CatImageProcessor

Наследуется от ImageProcessing. Отвечает за параллельную обработку изображений котов. Содержит метод edge_detection_cat для обработки одного изображения и process_images_parallel для параллельной обработки списка изображений через ProcessPoolExecutor.

2.4 Модуль cat_api_client.py

2.4.1 Класс CatAPIClient

Инкапсулирует функционал взаимодействия с Cat API. Реализует асинхронные методы:

1. fetch_cats_urls() – получение списка URL изображений
2. download_image() – загрузка одного изображения
3. download_images() –параллельная загрузка нескольких изображений

2.5 Модуль logger.py

2.5.1 Класс PipelineLogger

Настраивает логирование с двумя обработчиками (файл и консоль). Содержит декоратор timeit для замера времени выполнения асинхронных функций.

2.6 Модуль cat_pipeline.py

Главный модуль приложения, объединяющий все компоненты. Функция run_pipeline_async() реализует полный пайплайн: получение URL, скачивание, сохранение оригиналов, параллельная обработка, сохранение обработанных изображений.

2.7 Покрытие тестами

Используется встроенный модуль unittest для создания автоматических тестов. Для тестирования асинхронных функций применяется unittest.IsolatedAsyncioTestCase. Mock-объекты из unittest.mock позволяют тестировать API клиент без реальных сетевых запросов. Тестами покрыты классы CatImage, CatAPIClient, ImageProcessor.

2.8 Сетевое взаимодействие

Для эффективной работы с сетевыми запросами и файловыми операциями используется асинхронность. Библиотека aiohttp позволяет выполнять одновременные HTTP-запросы к Cat API, что значительно ускоряет загрузку нескольких изображений. Библиотека aiofiles обеспечивает неблокирующую запись изображений на диск.

Для обработки изображений используется ProcessPoolExecutor из модуля concurrent.futures. Это позволяет распределить вычислительно сложные операции свёртки по нескольким ядрам процессора, обходя ограничения GIL (Global Interpreter Lock) в Python

2.9 Основные используемые библиотеки

Здесь приведена часть библиотек, использовавшихся в ходе написания лабораторных работ.

1. aiofiles – для асинхронной записи в файл.
2. aiohttp – для асинхронного обращения к API.
3. pillow – для сохранения изображений.
4. numpy – для реализации кастомных методов работы с изображениями.
5. opencv-python – для обработки изображений и сравнения с кастомной реализацией алгоритмов.
6. asyncio – для работы с асинхронными операциями.

2.10 Инструкция по запуску

Далее приведена последовательность действий по запуску программы.

1. Клонировать репозиторий: git clone https://github.com/s44w/6401prokhorovsa.git
2. Создать виртуальное окружение: python -m venv .venv
3. Активировать виртуальное окружение: .venv\Scripts\activate
4. Установить зависимости: pip install -r requirements.txt
5. Создать файл .env в корне проекта с полями API_KEY и BASE_URL=https://api.thecatapi.com/v1/images/search
6. Запустить основной пайплайн обработки изображений: python cat_pipeline.py
7. При желании запустить тесты: python -m unittest discover tests

Если все настроено правильно, то после запуска программы информация о ее работе будет выводиться в консоль. Пример вывода представлен на рисунке 1.

```
2025-12-15 21:52:24,375 - INFO - Starting 'run_pipeline_async'...
2025-12-15 21:52:24,375 - INFO - Fetching cat URLs...
2025-12-15 21:52:25,682 - INFO - Downloading images...
2025-12-15 21:52:27,297 - INFO - Saving original images...
2025-12-15 21:52:27,667 - INFO - Processing images...
2025-12-15 21:52:27,667 - INFO - Starting parallel image processing...
2025-12-15 21:52:29,822 - INFO - Convolution for image 1 started (PID 11876)
2025-12-15 21:52:29,852 - INFO - Convolution for image 2 started (PID 13084)
2025-12-15 21:52:29,944 - INFO - Convolution for image 3 started (PID 1748)
2025-12-15 21:52:29,951 - INFO - Convolution for image 4 started (PID 30908)
2025-12-15 21:52:29,970 - INFO - Convolution for image 1 finished (PID 11876)
2025-12-15 21:52:29,979 - INFO - Convolution for image 5 started (PID 11876)
2025-12-15 21:52:30,052 - INFO - Convolution for image 5 finished (PID 11876)
2025-12-15 21:52:30,087 - INFO - Convolution for image 4 finished (PID 30908)
2025-12-15 21:52:30,313 - INFO - Convolution for image 2 finished (PID 13084)
2025-12-15 21:52:30,470 - INFO - Convolution for image 3 finished (PID 1748)
2025-12-15 21:52:30,894 - INFO - Parallel image processing completed
2025-12-15 21:52:30,894 - INFO - Saving processed images...
2025-12-15 21:52:31,134 - INFO - Pipeline completed. Saved 5 images to data
2025-12-15 21:52:31,138 - INFO - Finished 'run_pipeline_async' in  6.763s
```

Рисунок 1 – Пример работы программы

После этого в папке data будут сохранены изображения до и после обработки. Пример представлен на рисунке 2.

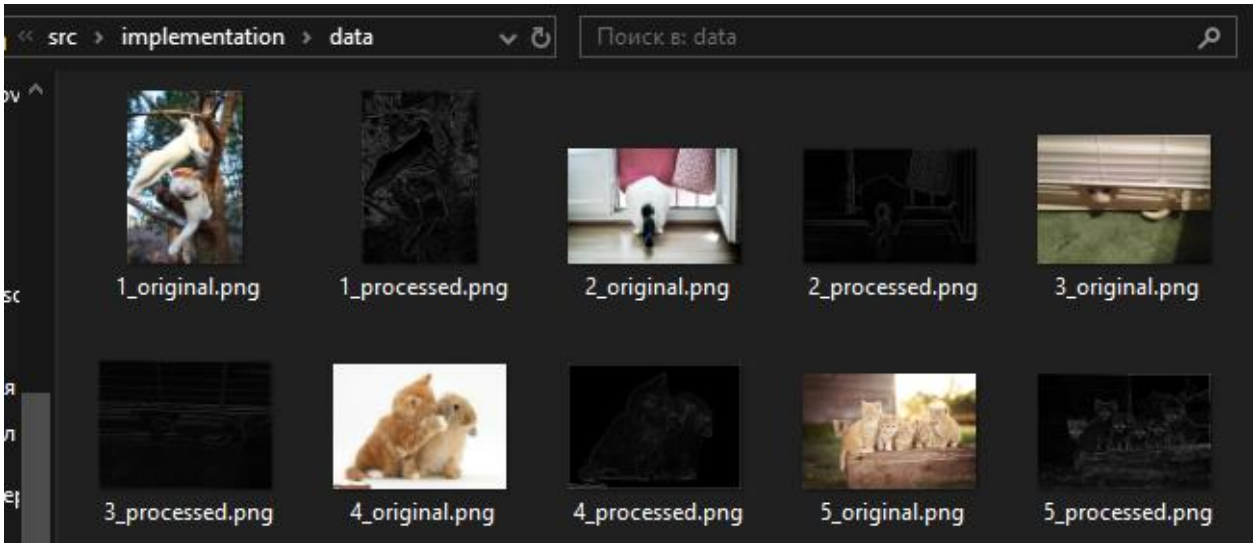

Рисунок 2 – Пример сохранения изображений

3 Заключение

В ходе выполнения лабораторных работ было создано полнофункциональное приложение для обработки изображений с использованием внешнего API. Успешно освоены следующие технологии и методы: асинхронность и многопроцессорность, алгоритмы обработки изображений, логирования в многопроцессорном коде, unit-тесты с использованием Mock-объектов.

Приложение А – ссылка на репозиторий GitHub

Приложение размещено по ссылке: https://github.com/s44w/6401prokhorovsa.git.

Приложение Б – основные зависимости проекта

Далее перечислены основные зависимости проекта.

1. aiofiles>=23.0.0
2. aiohttp>=3.9.0
3. pillow>=10.0.0
4. numpy>=1.24.0
5. scipy>=1.11.0
6. opencv-python>=4.8.0
7. python-dotenv>=1.0.0